Inhaltsverzeichnis

6 Vorwort

8 Vorstellung der Autoren

9 Anleitung

10 Kapitel 1: Einstellungen
Kommentar
11 Arbeitsblätter

13 Kapitel 2: So funktioniere ich
Kommentar
Arbeitsblätter

14 Kapitel 3: Teste mich!
Kommentar
Arbeitsblätter

15 Kapitel 4: Programme im Editor schreiben und sie speichern
Kommentar
16 Arbeitsblätter

21 Kapitel 5: Jetzt geht es los
Kommentar
22 Arbeitsblätter

25 Kapitel 6: Der Befehl repeat
Kommentar
26 Arbeitsblätter

28 Kapitel 7: Die Funktion repeat 2
Kommentar
Arbeitsblätter

30 Kapitel 8: Einstieg Parameter
Kommentar
Arbeitsblätter

32 Kapitel 9: Winkel in Vielecken
Kommentar
Arbeitsblätter

35 Kapitel 10: Weiterführung Parameter
Kommentar
36 Musterlösung
37 Arbeitsblätter

40 Kapitel 11: Gruppierungen
Kommentar
Mögliche Lösungen
42 Arbeitsblätter

44 Kapitel 12: Zählen 1 (Variablen)
Kommentar
45 Arbeitsblätter

49 Kapitel 13: Entscheidungen
Kommentar
Arbeitsblätter

52 Weitere Publikationen

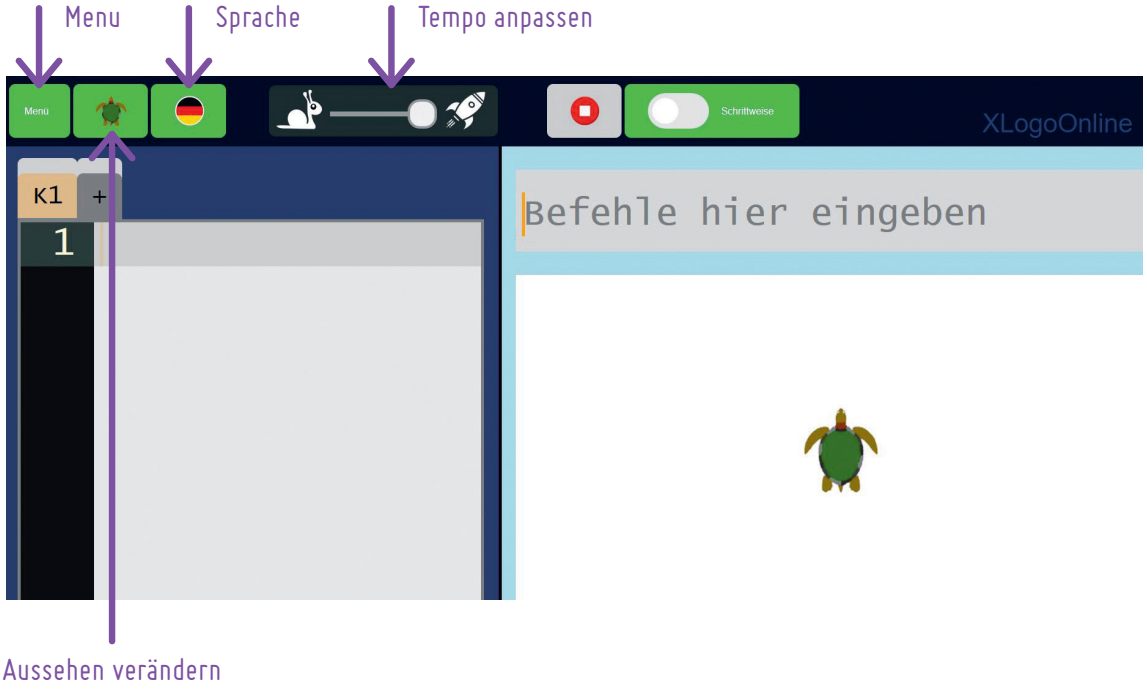
54 Eigene Notizen

Kapitel 1: Einstellungen



Darf ich mich vorstellen?
Ich bin XLogo, die Schildkröte. Ich werde dich beim Programmieren begleiten. Ich stelle dir Aufgaben, gebe dir aber auch Tipps, damit du sie lösen kannst. Lies immer genau, was ich dir sage, dann wirst du sicher Erfolg, aber vor allem auch Spass haben.

Ich empfehle dir, folgende Programmeinstellungen vorzunehmen:
Bei Menu kannst du unter anderem bestimmen, wie schnell ich zeichnen soll. Ich schlage dir vor, am Anfang die Tempomarkierung ganz nach links zu schieben.
Wenn du auf die Schildkröte klickst, kannst du mein Aussehen verändern. Wähle Englisch als Programmiersprache, denn so arbeiten die Programmierer auf der ganzen Welt.



Kapitel 2: So funktioniere ich

KOMMENTAR

Logo ist eine Programmiersprache, die es ermöglicht, die Schildkröte zu steuern.

1 Im Wesentlichen bewegt sie sich vorwärts (forward) oder rückwärts (back). Die Distanz, die sie zurücklegen soll, wird in Schritten (z. B. 100) definiert.

2 Soll sie sich drehen, muss die Richtung (right oder left) angegeben werden, und natürlich der gewünschte Winkel (z. B. 90).

3 Zu den meisten Befehlen gibt es auch eine abgekürzte Schreibeweise, z. B. fd für forward. In diesem Lehrmittel wird konsequent die ausgeschriebene Variante verwendet. Im für weiterführende Aufgaben empfohlenen Buch von Professor Juraj Hromkovič werden meistens die Abkürzungen benutzt.



Unten siehst du, wie ich mich bewegen kann:

`forward` 

`right` 

`left` 

Du sagst mir, wie viele Schritte ich machen soll, z. B. `forward 100`



und ich gehe dann genau 100 Schritte vorwärts.

Möchtest du, dass ich die Richtung ändere, befiehlst du mir, wie viele Grad ich mich drehen soll, z. B. `right 90`







2. Und wie sieht es mit dieser Aufgabe aus? Versuche das Bild wieder zuerst von Hand zu zeichnen, bevor du es programmierst. Nenne das Programm **exercise2**.

```
forward 100
right 90
forward 260
right 90
forward 80
left 90
forward 100
right 90
```

Wenn du die Aufgaben 1 und 2 geschafft hast, bist du bereit für den nächsten Schritt.
Kannst du dieses Bild programmieren?

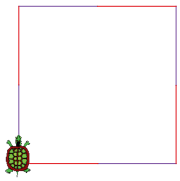
200

Tippe es in den Editor und nenne es **exercise3**.
Wenn es richtig ist, sichere es im Menu unter Save File.

Kapitel 13: Entscheidungen



Kannst du auch das?



Und ob du's glaubst oder nicht, ich kann sogar richtig rechnen. Ich kann dir nämlich sagen, ob eine Division aufgeht, oder ob sie Rest gibt.

Ich habe unten ein Programm für dich notiert und gebe dir dazu drei Aufträge:
1. Suche die Stelle, wo ich etwas berechne. Erkläre, was ich da bestimme! (Unterstreiche mit Rot!)
2. An welcher Stelle entscheide ich mich für etwas? (Unterstreiche mit Gelb!)

```
to zaun
      make "z1 1
      repeat 20 [
            make "rest mod :z1 5
            if :rest = 0 [
                  setpencolor blue
                  forward 180
                  left 180
                  forward 180
                  left 180
            ]     [
                  left 180
                  setpencolor red
                  forward 50
                  left 180
                  forward 50
            ]
            right 90
            forward 20
            left 90
            make "z1 :z1 + 1
      ]
end
```